UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/896,526	06/28/2001	Haitham Akkary	42390P11201	8421

8791 7590 05/18/2004

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD, SEVENTH FLOOR
LOS ANGELES, CA 90025

EXAMINER	
GOLE, AMOL V	
ART UNIT	**PAPER NUMBER**
2183	4

DATE MAILED: 05/18/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/896,526	Applicant(s) AKKARY ET AL.
	Examiner Amol V. Gole	Art Unit 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 6/28/01,2/10/03,3/18/03.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-35 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-35 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 28 June 2001 is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 2,3.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-35 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file:

#2: IDS (2/10/03)

#3: IDS (3/18/03)

Drawings

3. The drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Therefore, the

a) load thread id portion in claim 7

b) store thread id and store instruction validity portions in claim 8

c) array with head and tail pointers, wrap around bit in claim 9

d) limitations of claim 10

3) method steps in claims 14-19

must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

4. The drawings are objected to because in fig. 7, in block 753, the "trace" is misspelled as "trade". A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

Specification

5. The disclosure is objected to because of the following informalities:

a) It is not very clear how to spawn the speculative thread and how value prediction is implemented.

b) Para. 26 line 4 discloses that loads are posted into load buffer 150. It is not clear whether the loads are from the commit processor, the speculative processor or both.

c) Also, paragraph 29 is very confusing. The "store valid bit" is in both the store buffer and load buffer. Hence, it is not clear always clear which one is being referred to. Also it is difficult to understand what replayed instructions are. Are instructions replayed in the commit processor or speculative processor? Furthermore, in lines 6-8 it is disclosed that if a "replayed store" (which is being understood as a store being re-executed) has a matching store ID (presumably in

the load buffer) the store valid bit is cleared and a mispredicted bit is set in the trace buffer. What is not understood is when can a "replayed store" have the same store ID and why does it clear the valid bit and set a mispredicted bit? Is this misprediction with regard to the value prediction? How does this allow for the proper functioning of the processor? Moreover while discussing flushing the pipeline, it is not clear to which processor pipeline is being addressed. Also the conditions for flushing the pipeline are also not clear because it is not known whether the load in question is in the commit or speculative processor.

d) Also, it is disclosed clearly how it is found out whether or not speculative results are correct or not.

e) Para. 33 discloses that "replay mode" execution starts at the first instruction of a speculative thread. It is unclear whether the instructions are replayed in the commit processor or speculative processor.

f) The various headings of each section in the specification (e.g. Field of Invention, Background of the Invention, etc.) should not be underlined. Please make the appropriate changes.

Appropriate correction is required.

6. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required: Claim 16 discloses the step of "flushing a pipeline,... if one of the load is not replayed and does not match a tag portion, **and** the load instruction matches the tag portion in the load buffer ..". However, the specification in para. 29

states to flush the pipeline upon a not replayed load not matching any tag, **or** a load matching tag with the store valid bit clear.

7. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: A MULTI-PROCESSOR SYSTEM FOR MULTI-THREADED EXECUTION CAPABLE OF EXECUTING A SINGLE THREAD ON TWO PROCESSORS WHEREIN ONE PROCESSOR EXECUTES THE SAME SINGLE THREAD SPECULATIVELY AHEAD OF THE OTHER AND FEEDS CONTROL INFORMATION AND RESULTS TO THE OTHER FOR COMMITMENT RESULTING IN OVERALL SPEED UP IN EXECUTION.

Claim Objections

8. Claim 4 and 32 are objected to because of the following informalities: On line 2 of both the claims, it is claimed that the L0 data cache devices have exact copies of "data cache instructions". However this appears to be incorrect because the data cache contains data and not instructions. Hence The Office believes that the claim should be corrected to reflect this.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

9. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

10. Claims **14, 15, and 18** are rejected under 35 U.S.C. 112, first paragraph, as based on a disclosure which is not enabling. Knowing what a replayed instruction is and in which processor it is executed is critical or essential to the practice of the invention, but not included in the claim(s) is not enabled by the disclosure. See *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976).

On lines 3 and 4 of claim 14, it is disclosed to check the condition of whether a replayed store instruction has a matching store ID portion; on lines 2 and 3 of claim 15 it is disclosed to check the condition of whether a store instruction that is not replayed matches a store ID portion; on lines 3, 5, 8 of claim 18 it is disclosed to take certain actions with respect to replayed and non-replayed instructions. Firstly it is not disclosed in the claims or disclosure what a replayed store instruction is (e.g. is it part of a loop or is it re-executed by the same or other processor because of a misprediction?) and in which processor it is being executed without the knowledge of which it would not be

possible to make and use the invention. Furthermore with respect to claims 14 and 15, in order to make and use the invention without undue experimentation, one would need to know which store ID portion to compare it with i.e. the store ID in the store buffer or the load buffer.

11. Claim **16** is rejected under 35 U.S.C. 112, first paragraph, as based on a disclosure which is not enabling. Knowing which load is to be checked whether it is replayed or not and whether it matches a tag portion or not is critical or essential to the practice of the invention, but not included in the claim(s) is not enabled by the disclosure. See *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976).

On line 2 of claim **16**, the limitation of "one of the load..." is disclosed however it is not clear which load is being referred to because there may be many loads in a program before or after the load to be restarted. In order to make and use the invention without undue experimentation one would need to know which loads to consider and how to consider them when reading the limitation of "one of the load".

12. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

13. Claims **16 and 17** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

14. Claim 16 recites the limitation "one of the load" in line 4. There is insufficient antecedent basis for this limitation in the claim. Prior to "one of the load" only one load instruction was disclosed ("restarting a load instruction") while "one of the load" refers to one of a plurality of load instructions.

15. Regarding claim 17, the phrase "approaching an empty state" renders the claim indefinite because the specification lacks some standard for measuring the degree intended.

Claim Rejections - 35 USC § 102

16. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

17. Claims **1-7, 11-13, 17,19** are rejected under 35 U.S.C. 102(a) as being anticipated by Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000).

18. In regard to claim 1:

19. Sudaramoorthy et al. disclose an apparatus comprising:

a first processor and a second processor (fig. 1, R-stream and A-stream processors. Each processor comprising the execute core);

a plurality of memory devices coupled to the first processor and the second processor (fig. 1, I-cache and D-cache memories);

a register buffer coupled to the first processor and the second processor (fig .1, the delay buffer serves as a register buffer when an IR-misprediction is detected because the register values are copied from the R-stream processor to the A-stream

processor via the delay buffer [as shown by the dotted lines in fig. 1 and col. 12, lines 7-12]);

a trace buffer coupled to the first processor and the second processor (fig. 1; col. 10, lines 17-19; col. 11, lines 7-17: the delay buffer during normal operation serves as a trace buffer because the control and data information of instructions traced is passed from the A-stream to the R-stream processor); and

a plurality of memory instruction buffers coupled to the first processor and the second processor (fig. 1, a separate reorder buffer is connected to each processor);

wherein the first processor and the second processor perform single threaded applications using multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single thread is instantiated twice to create two threads and each thread is run on different processors).

20. In regard to claim 2:

21. Sudaramoorthy et al. disclose the apparatus of claim 1, wherein the memory devices comprise of a plurality of cache devices (Fig. 1, I-Cache and D-Cache).

22. In regard to claim 3:

23. Sudaramoorthy et al. disclose the apparatus of claim 1, wherein the first processor is coupled to at least one of a plurality of zero level (L0) data cache devices and at least one of a plurality of L0 instruction cache devices, and the second processor is coupled to at least one of the plurality of L0 data cache devices and at least one of

the plurality of L0 instruction cache devices (fig. 1 shows that each processor is connected to a separate data cache (D-Cache) and instruction (I-Cache) which can be considered as zero-level caches because they are directly connected to the execute cores).

24. In regard to claim 4:

25. Sudaramoorthy et al. disclose the apparatus of claim 3, wherein each of the plurality of L0 data cache devices having exact copies of data cache instructions, and each of the plurality of L0 instruction cache devices having exact copies of instruction cache instructions (Although this is not mentioned explicitly, it is deemed inherent to the design because as each processor is executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor must contain exact copies of instructions and data).

26. In regard to claim 5:

27. Sudaramoorthy et al. disclose the apparatus of claim 1, wherein the plurality of memory instruction buffers includes at least one store forwarding buffer (fig. 1,reorder buffer connected to A-stream processor) and at least one load-ordering buffer (fig. 1, reorder buffer connected to R-stream processor).

28. In regard to claim 6:

29. Although Sudaramoorthy et al. do not mention that the at least one store forwarding buffer (fig. 1, reorder buffer (ROB) connected to A-stream processor) comprises a structure having a plurality of entries, each of the plurality of entries having a tag portion, a validity portion, a data portion, a store instruction identification (ID) portion, and a thread ID portion it is deemed inherent to the design. A ROB is used to order instructions completing execution hence must contain a plurality of entries. Also each entry must have a tag portion to index into the ROB, a validity portion to indicate whether an entry can be written to or read from, a data portion for storing the results of the instruction, a store instruction ID portion would be the instruction opcode of an entry, and a thread ID for indicating which thread that instruction belongs to.

30. In regard to claim 7:

31. Although Sudaramoorthy et al. do not mention that the at least one load ordering buffer (fig. 1, reorder buffer connected to R-stream processor) comprises a structure having a plurality of entries, each of the plurality of entries having a tag portion, an entry validity portion, a load identification (ID) portion, and a load thread ID portion it is deemed inherent to the design. A ROB is used to order instructions completing execution hence must contain a plurality of entries. Also each entry must have a tag portion to index into the ROB, a validity portion to indicate whether an entry can be written to or read from, a load instruction ID portion would be the instruction opcode of an entry, and a thread ID for indicating which thread that instruction belongs to.

32. **In regard to claim 11:**

33. Sundaramoorthy et al. disclose a method comprising:

executing a plurality of instructions in a first thread by a first processor (col. 1, lines 53-54, col. 2, lines 18-23: The R-stream thread is executed by the R-stream processor in fig. 1. The R-stream processor comprises of the execute core and the IR-detector and IR-predictor); and

executing the plurality of instructions in the first thread by a second processor (col. 1, lines 53-54, col. 2, lines 18-32: The A-stream thread, which is the same as the R-stream thread, is executed by the A-stream processor) as directed by the first processor (col. 4, lines 21-38: IR-detector and IR-predictor in fig. 1, which are part of the first processor i.e. R-stream processor, direct the second processor (A-stream processor) to execute instructions from the A-stream), the second processor executing the plurality of instructions ahead of the first processor (col. 2, lines 20-23: A-stream runs ahead of the R-stream and it is executed by the second processor (A-stream processor)).

34. **In regard to claim 12:**

35. Sundaramoorthy et al. disclose the method of claim 11, further including:

transmitting control flow information from the second processor to the first processor, the first processor avoiding branch prediction by receiving the control flow information (col. 10, lines 17-21, 30-35, 43-46); and

transmitting results from the second processor to the first processor, the first processor avoiding executing a portion of instructions (col. 10, lines 17-21, 30-33, 35-38: results (data-flow information) are transmitted from the A-stream processor to the R-stream processor via the delay buffer, and these values are used directly by the instructions hence avoiding the execution of the portion of the instructions) by committing the results of the portion of instructions into a register file from a trace buffer (Although this is not explicitly mentioned, it is deemed inherent to the design because col. 4 line 15 discloses the presence of a register file in the processor and as results are written to the register file so that they can be read from by future instructions, the results of the instructions from the trace buffer (delay buffer) must be written into the register file).

36. In regard to claim 13:

37. Sundaramoorthy et al. disclose the method of claim 12, further including: duplicating memory information in separate memory devices for independent access by the first processor and the second processor (Although this is not mentioned explicitly, it is deemed inherent to the design because as each processor is executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor (fig. 1) must contain exact copies of instructions and data).

38. In regard to claim 17:

39. Sundaramoorthy discloses the method of claim 12, further including:

executing a replay mode at a first instruction of a speculative thread (col. 1, lines 53-54, col. 2, lines 18-20: This feature is deemed inherent to the reference because when the A-stream is initially started i.e. at the first instruction, there will be two redundant threads being executed which means the thread is being replayed from that point. This can be called a replay mode); terminating the replay mode and the execution of the speculative thread if a partition in the trace buffer is approaching an empty state (this limitation is also deemed inherent to the reference because when the partition in the trace buffer (delay buffer col. 10 lines 15+) is approaching an empty state that means the A-stream has stopped producing results and finished executing. Therefore now the replay mode and the A-stream are terminated).

40. In regard to claim 19:

41. Sundaramoorthy teaches the method of claim 12, further including:

clearing a valid bit in an entry in a load buffer (fig. 1, the reorder buffer connected to the R-stream processor) if the load entry is retired (Although not explicitly mentioned, it is deemed inherent to the design because a load entry, on being retired, has to be marked invalid to ensure that other new instructions can occupy that entry safely).

Claim Rejections - 35 USC § 103

42. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> *(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

43. Claims **8-10** rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000).

44. In regard to claim 8:

45. Sundaramoorthy et al. do not disclose the apparatus of claim 7, wherein each of the plurality of entries further having a store thread ID portion, a store instruction ID portion, and a store instruction validity portion.

46. However this difference is only found in the nonfunctional descriptive material and are not involved in the functioning of the load ordering buffer. Thus this descriptive material will not distinguish the claimed invention from the prior art in terms of

patentability, see *In re Gulack*, 703 F.2d 1381, 1385, 217 USPQ 401, 404 (Fed. Cir. 1983); *In re Lowry*, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994).

47. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have these portions in the load ordering buffer (ROB connected to the R-stream processor) of Sudaramoorthy because such portions do not alter how the load ordering buffer functions and because the subjective interpretation of the portions does not patentably distinguish the claimed invention from prior art. See *Gulack* cited above.

48. In regard to claim 9:

49. Although Sundaramoorthy et al. discloses that the trace buffer (delay buffer) is a FIFO queue (col. 10, line 17), they do not disclose that the trace buffer is a circular buffer having an array with head and tail pointers, the head and tail pointers having a wrap-around bit.

50. "Official Notice" is taken that it is well known and expected in the art to implement a FIFO queue as a circular buffer with head and tail pointers wherein head and tail pointers have a wrap-around bit. A circular buffer is useful to implement in hardware because only the head and tail pointers need to be incremented/decremented instead of actually physically shifting entries. A wrap around bit would also be needed to indicate whether the pointer has wrapped around the end of the queue.

51. Therefore, it would been obvious to one of ordinary skill in the art at the time of the invention to have implemented the FIFO queue as a circular buffer with head and

tail pointers, the head and tail pointers having a wrap around bit because it is known that a FIFO queue can be implemented as a circular buffer and it is easier to build in hardware.

52. In regard to claim 10:

53. apparatus of claim 1, the register buffer comprising an integer register buffer and a predicate register buffer.

54. Sundaramoorthy et al. do not disclose the apparatus of claim 1, the register buffer comprising an integer register buffer and a predicate register buffer.

55. However this difference is only found in the nonfunctional descriptive material and are not involved in the functioning of the register buffer. Thus this descriptive material will not distinguish the claimed invention from the prior art in terms of patentability, see *In re Gulack*, 703 F.2d 1381, 1385, 217 USPQ 401, 404 (Fed. Cir. 1983); *In re Lowry*, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994).

56. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have an integer register buffer and a predicate register buffer in the register buffer (delay buffer) of Sudaramoorthy because such portions do not alter how the register buffer functions and because the subjective interpretation of the portions does not patentably distinguish the claimed invention from prior art. See *Gulack* cited above.

57. Claims **14-16** are rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000) in view of Akkary (WO 99/31594).

58. In regard to claim 14:

59. Sundaramoorthy et al. do not disclose clearing a store validity bit and setting a mispredicted bit in a load entry in the trace buffer if a replayed store instruction has a matching store identification (ID) portion.

60. "Official Notice" is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

61. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). He discloses clearing a store validity bit (SB Hit field) in the load buffer if data came from memory (pg. 37, para. 3, line 4; pg. 38, line 1). Also when a store instruction is executed (which includes replayed stores), its address is compared with the store ID portion (addresses) of load instructions (pg. 36, para. 3). On a match, a replay event is signaled to the load entry in the trace buffer to replay the load instruction and all its dependant instructions because it was mispredicted (pg. 38, para. 2).

62. "Official Notice" is taken that is well known and expected in the art to set a status bit to indicate the change of state. e.g. setting a mispredicted bit when an instruction associated with that bit is signaled to be mispredicted.

63. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by clearing a store validity bit and setting a mispredicted bit in a load entry in the trace buffer (delay buffer) if a replayed store instruction has a matching store ID portion.

64. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

65. In regard to claim 15:

66. Sundaramoorthy et al. do not teach setting a store validity bit if a store instruction that is not replayed matches a store identification (ID) portion.

67. “Official Notice” is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

68. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). He discloses setting a store validity bit (SB Hit field) in the load buffer if data came from store buffer (pg. 37, para. 3, line 4; pg. 38, lines 1-2). In order for data to come from the store buffer, a store

instruction address (including store instructions that are not replayed) must match a store ID portion (address) of the load entry.

69. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by setting a store validity bit if a store instruction that is not replayed matches the store ID portion.

70. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

71. In regard to claim 16:

72. Although Sundaramoorthy et al. disclose flushing the pipeline (reorder buffer) of the R-stream on a misprediction, they do not disclose flushing a pipeline, setting a mispredicted bit in a load entry in the trace buffer and restarting a load instruction if one of the load is not replayed and does not match a tag portion in a load buffer, and the load instruction matches the tag portion in the load buffer while a store valid bit is not set.

73. "Official Notice" is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

74. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). In particular, when a store valid bit is not set (SB hit = 0, pg. 38, para. 2) and when a store instruction compared with the addresses of load instructions (pg. 36, para. 3) is a match, a replay event is signaled to the load entry in the trace buffer to replay the load instruction and all its dependant instructions because it was mispredicted (pg. 38, para. 2).

75. "Official Notice" is taken that is well known and expected in the art to set a status bit to indicate the change of state. e.g. setting a mispredicted bit when an instruction associated with that bit is signaled to be mispredicted.

76. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment and flush the pipeline on reading the mispredicted bit. Therefore it would have been obvious to one ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by flushing a pipeline, setting a mispredicted bit in a load entry in the trace buffer and restarting a load instruction if one of the load is not replayed and does not match a tag portion in a load buffer, and the load instruction matches the tag portion in the load buffer while a store valid bit is not set.

77. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

78. Claim **18** is rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000) in view of Hennessy and Patternson ("Computer Architecture A Quantitative Approach", Morgan Kaufmann, 1996).

79. In regard to claim 18:

80. Sundaramoorthy discloses

issuing all instructions up to a next replayed instruction including dependent instructions (This feature is deemed inherent to the design because in order to execute the thread all instructions are issued in either one of the R-stream and A-stream processors);

issuing instructions that are not replayed as no-operation (NOPs) instructions (This feature is also deemed inherent to the design because if an instruction that is not replayed does not occupy a slot in the execution pipeline it will lead to improper functioning of the processor. Hence as the instruction that is not replayed is not to be executed, a NOP must be issued in its place);

issuing all load instructions and store instructions to memory (This limitation is also deemed inherent to the design because if all loads and stores are not issued to memory, the state of the thread would be incorrect leading to the malfunctioning of the system);

committing non-replayed instructions from the trace buffer to the register file (Although this is not explicitly mentioned, it is deemed inherent to the design because col. 4 line 15 discloses the presence of a register file in the processor and as results are written to the register file so that they can be read from by future instructions, the results of the instructions from the trace buffer (delay buffer) that are not going to be replayed must be written into the register file).

81. However Sundaramoorthy et al. do not teach supplying names from the trace buffer to preclude register renaming.

82. Hennessy and Patterson teach that register renaming is used to reduce name dependencies allowing instructions involved in name dependencies to execute simultaneously or be reordered (pg. 232, para. 5). As these dependencies are resolved, more instruction level parallelism can be extracted and performance can be improved.

83. One of ordinary skill in the art would have recognized to use register renaming in the Sundaramoorthy reference because it too would improve performance. As the trace buffer (delay buffer) would also supply the names, it would be logical not to do the renaming again in the R-stream processor. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by adding register renaming capabilities and supply names from the trace buffer to preclude register renaming.

One would have been motivated to do so because it would improve performance which is one of the objectives of the Sundaramoorthy reference (col. 1, lines 29-36).

84. Claims **20-22, 26, and 28** are rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000) in view of Tannenbaum ("Structured Computer Organization," Prentice-Hall, 1984, pp. 10-12).

85. In regard to claim 20:

86. Sundaramoorthy et al. disclose a method comprising:

executing a plurality of instructions in a first thread by a first processor (col. 1, lines 53-54, col. 2, lines 18-23: The R-stream thread is executed by the R-stream processor in fig. 1); and

executing the plurality of instructions in the first thread by a second processor (col. 1, lines 53-54, col. 2, lines 18-32: The A-stream thread, which is the same as the R-stream thread, is executed by the A-stream processor) as directed by the first processor (col. 4, lines 21-38: IR-detector and IR-predictor in fig. 1, considered part of the first processor i.e. R-stream processor, direct the second processor (A-stream processor) to execute instructions from the A-stream), the second processor executing the plurality of instructions ahead of the first processor (col. 2, lines 20-23: A-stream runs ahead of the R-stream and it is executed by the second processor (A-stream processor)).

87. However, Sundaramoorthy does not disclose an apparatus comprising a machine-readable medium containing instructions which, when executed by a machine to perform the method.

88. Tannenbaum teaches that any instruction executed by hardware can also be simulated in software (pg 11, para. 4, lines 1-2). He also teaches that hardware is generally immutable (first para. after sec. 1.4 header) while software allows for more rapid change (pg. 11, para. 4, lines 2-4).

89. One of ordinary skill in the art at the time of the invention would have been motivated to convert the Sundaramoorty et al. reference to software i.e. instructions on a machine readable medium because Tannebaum teaches that hardware is generally immutable (first para. after sec. 1.4 header) while software allows for more rapid change (pg. 11, para. 4, lines 2-4). Therefore, to allow for ease of correction of mistakes, and/or an ease of addition of new functionality, it would have been obvious to one of ordinary skill in the art to have implemented the method of Sundaramoorthy et al. by an apparatus comprising instructions recorded on a machine readable medium.

90. In regard to claim 21:

91. Sundaramoorthy et al. disclose the apparatus of claim 20, further including:

transmitting control flow information from the second processor to the first processor, the first processor avoiding branch prediction by receiving the control flow information (col. 10, lines 17-21, 30-35, 43-46); and

92. In regard to claim 22:

93. Sundaramoorthy et al. disclose the apparatus of claim 20, further including: duplicating memory information in separate memory devices for independent access by

the first processor and the second processor (Although this is not mentioned explicitly, it is deemed inherent to the design because as each processor is executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor (fig. 1) must contain exact copies of instructions and data).

94. In regard to claim 26:

95. Sundaramoorthy discloses the apparatus of claim 20, further including: executing a replay mode at a first instruction of a speculative thread (col. 1, lines 53-54, col. 2, lines 18-20: This feature is deemed inherent to the reference because when the A-stream is initially started i.e. at the first instruction, there will be two redundant threads being executed which means the thread is being replayed from that point. This can be called a replay mode); terminating the replay mode and the execution of the speculative thread if a partition in the trace buffer is approaching an empty state (this limitation is also deemed inherent to the reference because when the partition in the trace buffer (delay buffer col. 10 lines 15+) is approaching an empty state that means the A-stream has stopped producing results and finished executing. Therefore now the replay mode and the A-stream are terminated).

96. In regard to claim 28:

97. Sundaramoorthy teaches the apparatus of claim 20, further including:

clearing a valid bit in an entry in a load buffer (fig. 1, the reorder buffer connected to the R-stream processor) if the load entry is retired (Although not explicitly mentioned,

it is deemed inherent to the design because a load entry, on being retired, has to be marked invalid to ensure that other new instructions can occupy that entry safely).

98. Claims **23-25** are rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000) in view of Akkary (WO 99/31594).

99. In regard to claim 23:

100. Sundaramoorthy et al. do not disclose clearing a store validity bit and setting a mispredicted bit in a load entry in the trace buffer if a replayed store instruction has a matching store identification (ID) portion.

101. "Official Notice" is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

102. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). He discloses clearing a store validity bit (SB Hit field) in the load buffer if data came from memory (pg. 37, para. 3, line 4; pg. 38, line 1). Also when a store instruction is executed (which includes replayed stores), its address is compared with the store ID portion (addresses) of load instructions (pg. 36, para. 3). On a match, a replay event is signaled to the load entry in the trace buffer to replay the load instruction and all its dependant instructions because it was mispredicted (pg. 38, para. 2).

103. "Official Notice" is taken that is well known and expected in the art to set a status bit to indicate the change of state. e.g. setting a mispredicted bit when an instruction associated with that bit is signaled to be mispredicted.

104. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by clearing a store validity bit and setting a mispredicted bit in a load entry in the trace buffer (delay buffer) if a replayed store instruction has a matching store ID portion.

105. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

106. In regard to claim 24:

107. Sundaramoorthy et al. do not teach setting a store validity bit if a store instruction that is not replayed matches a store identification (ID) portion.

108. "Official Notice" is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

109. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). He discloses setting a store

validity bit (SB Hit field) in the load buffer if data came from store buffer (pg. 37, para. 3, line 4; pg. 38, lines 1-2). In order for data to come from the store buffer, a store instruction address (including store instructions that are not replayed) must match a store ID portion (address) of the load entry.

110. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by setting a store validity bit if a store instruction that is not replayed matches the store ID portion.

111. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

112. In regard to claim 25:

113. Although Sundaramoorthy et al. disclose flushing the pipeline (reorder buffer) of the R-stream on a misprediction, they do not disclose flushing a pipeline, setting a mispredicted bit in a load entry in the trace buffer and restarting a load instruction if one of the load is not replayed and does not match a tag portion in a load buffer, and the load instruction matches the tag portion in the load buffer while a store valid bit is not set.

114. "Official Notice" is taken that it is well known and expected in the art to use load and store buffers for the proper handling of memory operations.

115. Akkary discloses a system for ordering loads and stores in a multithreaded processor using load and store buffers (fig. 2, 182,184). In particular, when a store valid bit is not set (SB hit = 0, pg. 38, para. 2) and when a store instruction compared with the addresses of load instructions (pg. 36, para. 3) is a match, a replay event is signaled to the load entry in the trace buffer to replay the load instruction and all its dependant instructions because it was mispredicted (pg. 38, para. 2).

116. "Official Notice" is taken that is well known and expected in the art to set a status bit to indicate the change of state. e.g. setting a mispredicted bit when an instruction associated with that bit is signaled to be mispredicted.

117. One of ordinary skill in the art would have recognized that one could use the load and store buffer arrangement of Akkary in the Sundaramoorthy reference in order handle loads and stores in the multithreaded environment and flush the pipeline on reading the mispredicted bit. Therefore it would have been obvious to one ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by flushing a pipeline, setting a mispredicted bit in a load entry in the trace buffer and restarting a load instruction if one of the load is not replayed and does not match a tag portion in a load buffer, and the load instruction matches the tag portion in the load buffer while a store valid bit is not set.

118. One would have been motivated to do so because the Sundaramoorthy reference does not provide enough detail on how to handle loads and stores which

would lead one of ordinary skill in the art to seek a method of handling loads and stores in a multithreaded environment.

119. Claim **27** is rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000) in view of Hennessy and Patternson ("Computer Architecture A Quantitative Approach", Morgan Kaufmann, 1996).

120. In regard to claim 27:

121. Sundaramoorthy discloses

issuing all instructions up to a next replayed instruction including dependent instructions (This feature is deemed inherent to the design because in order to execute the thread all instructions are issued in either one of the R-stream and A-stream processors);

issuing instructions that are not replayed as no-operation (NOPs) instructions (This feature is also deemed inherent to the design because if an instruction that is not replayed does not occupy a slot in the execution pipeline it will lead to improper functioning of the processor. Hence as the instruction that is not replayed is not to be executed, a NOP must be issued in its place);

issuing all load instructions and store instructions to memory (This limitation is also deemed inherent to the design because if all loads and stores are not issued to

memory, the state of the thread would be incorrect leading to the malfunctioning of the system);

committing non-replayed instructions from the trace buffer to the register file (Although this is not explicitly mentioned, it is deemed inherent to the design because col. 4 line 15 discloses the presence of a register file in the processor and as results are written to the register file so that they can be read from by future instructions, the results of the instructions from the trace buffer (delay buffer) that are not going to be replayed must be written into the register file).

122. However Sundaramoorthy et al. do not teach supplying names from the trace buffer to preclude register renaming.

123. Hennessy and Patterson teach that register renaming is used to reduce name dependencies allowing instructions involved in name dependencies to execute simultaneously or be reordered (pg. 232, para. 5). As these dependencies are resolved, more instruction level parallelism can be extracted and performance can be improved.

124. One of ordinary skill in the art would have recognized to use register renaming in the Sundaramoorthy reference because it too would improve performance. As the trace buffer (delay buffer) would also supply the names, it would be logical not to do the renaming again in the R-stream processor. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Sundaramoorthy reference by adding register renaming capabilities and supply names from the trace buffer to preclude register renaming.

One would have been motivated to do so because it would improve performance which is one of the objectives of the Sundaramoorthy reference (col. 1, lines 29-36).

125. Claims **29-35** are rejected under 35 U.S.C. 103(a) as being unpatentable over Sundaramoorthy et al. ("Slipstream Processors: Improving both Performance and Fault Tolerance", ASPLOS, pp. 257-268, Nov. 2000).

126. In regard to claim 29:

127. A system comprising:

a first processor (fig. 1, R-stream processor comprising of the execute core);

a second processor (fig. 1, A-stream processor comprising of the execute core);

a bus coupled to the first processor and the second processor (fig. 1, a bus is shown between the first and second processors via the delay buffer);

a plurality of local memory devices coupled to the first processor and the second processor (fig. 1, I-cache and D-cache memories);

a register buffer coupled to the first processor and the second processor (fig .1, the delay buffer serves as a register buffer when an IR-misprediction is detected because the register values are copied from the R-stream processor to the A-stream processor via the delay buffer [as shown by the dotted lines in fig. 1 and col. 12, lines 7-12]);

a trace buffer coupled to the first processor and the second processor (fig. 1; col. 10, lines 17-19; col. 11, lines 7-17: the delay buffer during normal operation serves as a

trace buffer because the control and data information of instructions traced is passed from the A-stream to the R-stream processor); and

a plurality of memory instruction buffers coupled to the first processor and the second processor (fig. 1, a separate reorder buffer is connected to each processor);

wherein the first processor and the second processor perform single threaded applications using multithreading resources (col. 1, lines 53-54, col. 2, lines 18-20: teaches that a single thread is instantiated twice to create two threads and each thread is run on different processors).

128. Sundaramoorthy does not disclose a main memory coupled to the bus.

129. "Official Notice" is taken that it is well known and expected in the art to have a main memory connected to multiple processors via a common bus in a multi-processor environment.

130. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have added a main memory coupled to the bus in the Sundaramoorthy reference.

131. It would have been obvious to have a main memory because it is well known and expected and because a cache does not store all the data/instructions.

132. In regard to claim 30:

133. Sudaramoorthy et al. disclose the system of claim 29, wherein the memory devices comprise of a plurality of cache devices (Fig. 1, I-Cache and D-Cache).

134. In regard to claim 31:

135. Sudaramoorthy et al. disclose the system of claim 29, wherein the first processor is coupled to at least one of a plurality of zero level (L0) data cache devices and at least one of a plurality of L0 instruction cache devices, and the second processor is coupled to at least one of the plurality of L0 data cache devices and at least one of the plurality of L0 instruction cache devices (fig. 1 shows that each processor is connected to a separate data cache (D-Cache) and instruction (I-Cache) which can be considered as zero-level caches because they are directly connected to the execute cores).

136. In regard to claim 32:

137. Sudaramoorthy et al. disclose the system of claim 31, wherein each of the plurality of L0 data cache devices having exact copies of data cache instructions, and each of the plurality of L0 instruction cache devices having exact copies of instruction cache instructions (Although this is not mentioned explicitly, it is deemed inherent to the design because as each processor is executing the same thread (col. 1, lines 53-54, col. 2, lines 18-20) the instruction and data caches in each processor must contain exact copies of instructions and data).

138. In regard to claim 35:

139. Sundaramoorthy et al. does not teach the first processor and the second processor each sharing a first level (L1) cache device and a second level (L2) cache device.

140. "Official Notice" is taken that it is well known and expected in the art that a processors in a multi-processor environment share L1 and L2 cache devices.

141. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to have the first and second processors share L1 and L2 cache devices.

142. One would have been motivated to do so because it is well known and expected and simplifies cache coherency.

143. In regard to claim 34:

144. Sudaramoorthy et al. disclose the system of claim 31, wherein the plurality of memory instruction buffers includes at least one store forwarding buffer (fig. 1,reorder buffer connected to A-stream processor) and at least one load-ordering buffer (fig. 1, reorder buffer connected to R-stream processor).

145. In regard to claim 35:

146. Although Sudaramoorthy et al. do not mention that the at least one store forwarding buffer (fig. 1, reorder buffer (ROB) connected to A-stream processor) comprises a structure having a plurality of entries, each of the plurality of entries having

a tag portion, a validity portion, a data portion, a store instruction identification (ID) portion, and a thread ID portion it is deemed inherent to the design. A ROB is used to order instructions completing execution hence must contain a plurality of entries. Also each entry must have a tag portion to index into the ROB, a validity portion to indicate whether an entry can be written to or read from, a data portion for storing the results of the instruction, a store instruction ID portion would be the instruction opcode of an entry, and a thread ID for indicating which thread that instruction belongs to.

Conclusion

147. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty, which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections. See 37 CFR § 1.111.

a. E. Rotenberg et al. ("Trace processors", In Proceedings of the 30th Annual International Symposium on Microarchitecture, pages 138--148, December 1--3, 1997) teach how trace processors can aid in executing instructions more efficiently.

b. Lee et al. (US006629271B1) teach a ROB for a multithreaded processor (fig. 2)

c. Wang et al. (US 20020144083A1) teach speculatively spawning threads to precompute or prefetch data in advance of the main thread.

d. Tullsen et al. (Simultaneous multithreading: Maximizing on-chip parallelism. In Proceedings of the 22nd Annual International Symposium on Computer Architecture, pages 392--403, June 22--24, 1995) teach a method of simultaneous multithreading. Note section 5 regarding the cache design for SMTs in which the L2 and L3 caches are shared.

e. Roth and Sohi et al. ("Speculative Data-Driven Multithreading" In HPCA-7, Jan., 2001) teach a method of forking speculative threads that calculate critical computations ahead of time to speed up processing.

f. Mukherjee et al. (US006598122B2) discloses a redundantly threaded processor with a load buffer.

148. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Amol V. Gole whose telephone number is 703-305-8888. The examiner can normally be reached on 9:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 703-305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AVG
amol.gole@uspto.gov

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100